LA-UR-19-30136
Approved for public release; distribution is unlimited.

Title: Simplified Interface to Complex Memory (SICM) FY19 Project Review

Author(s): Lang, Michael Kenneth

Intended for: project review

Issued: 2019-10-07

Simplified Interface to Complex Memory (SICM) FY19 Project Review

September 2019 ECP ST Project Review

ECP Project FY20 WBS 2.3.1.16)

Business Sensitive Information

PI: Michael Lang (Los Alamos National Laboratory)

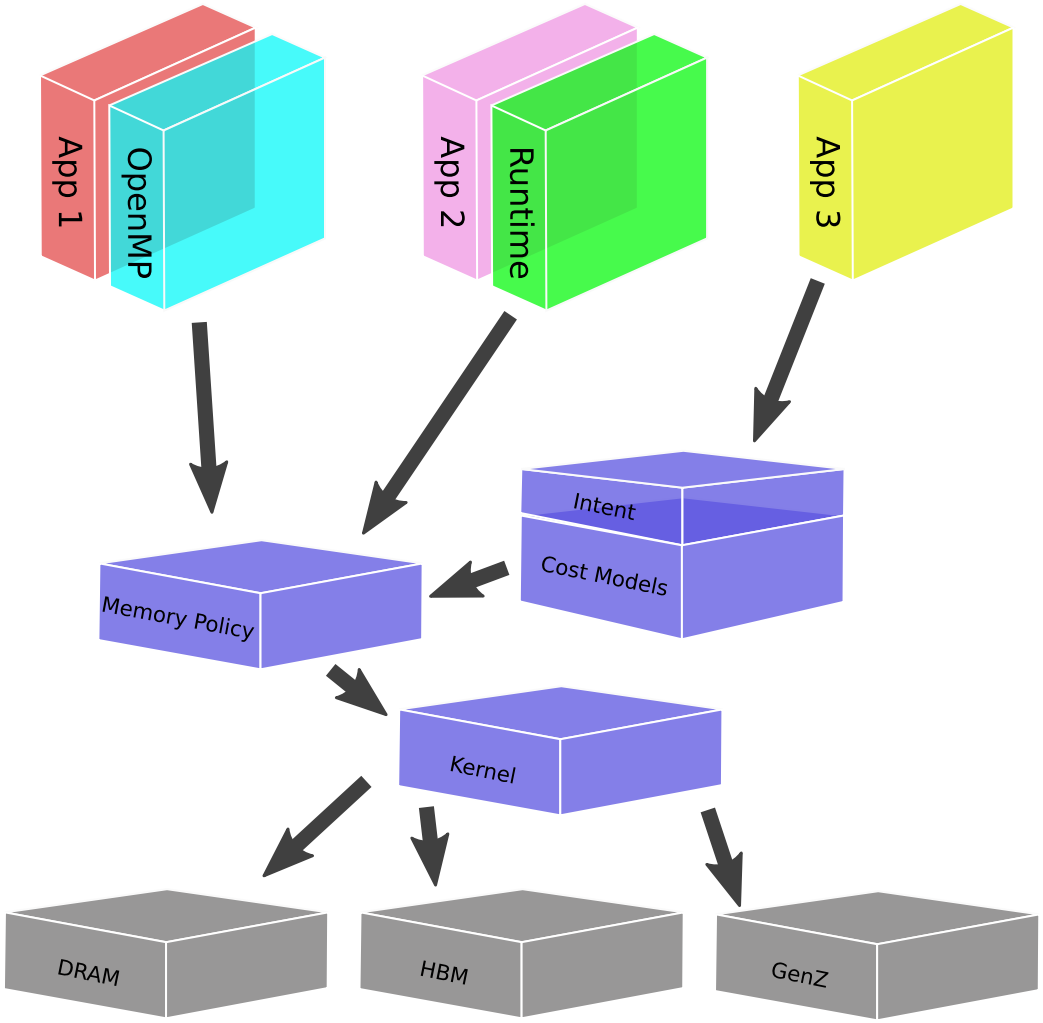
Date: Sept 27 2019

*(NNSA/ATDM funded)

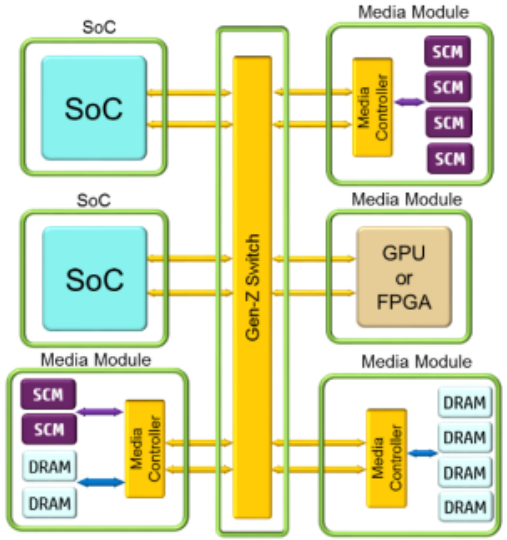
National Nuclear Security Administration

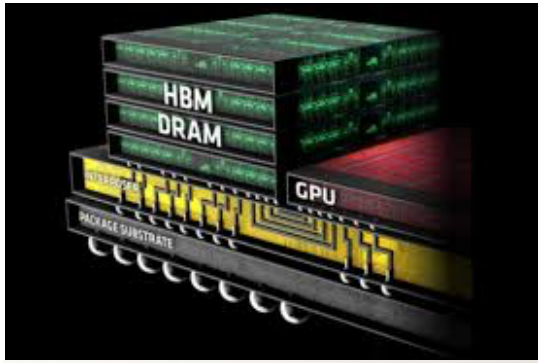
U.S. DEPARTMENT OF ENERGY | Office of Science

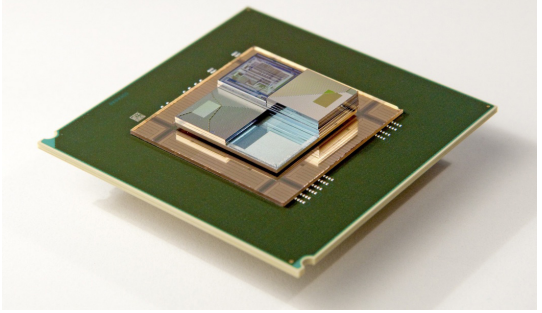
Simplified Interface to Complex Memory (SICM) Overview



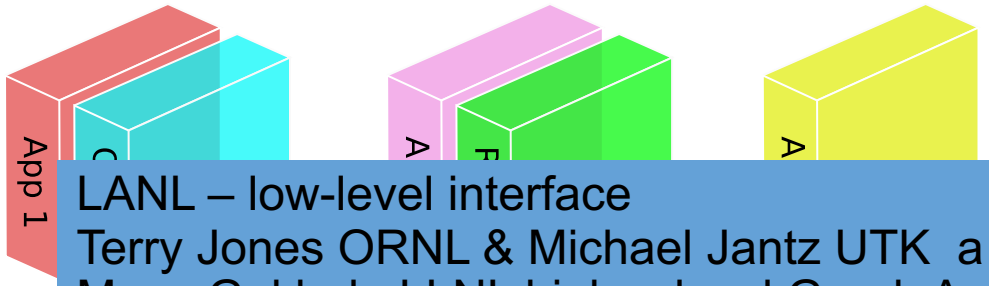
Allocate
Deallocate
Migrate
Arbitrate
&
Introspect
Memory
in a
portable
manner



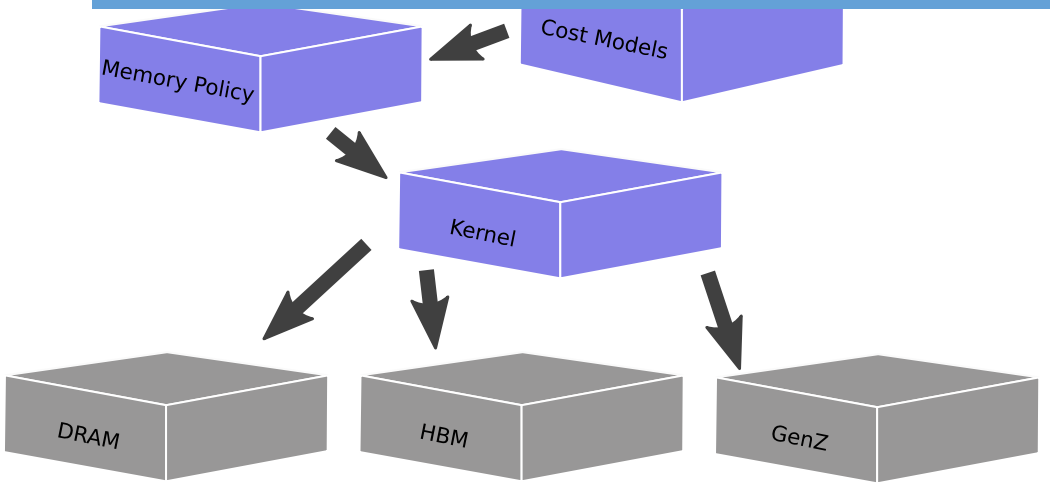




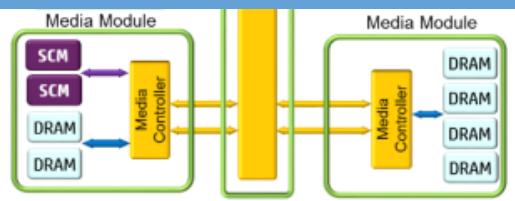
Simplified Interface to Complex Memory (SICM) Overview

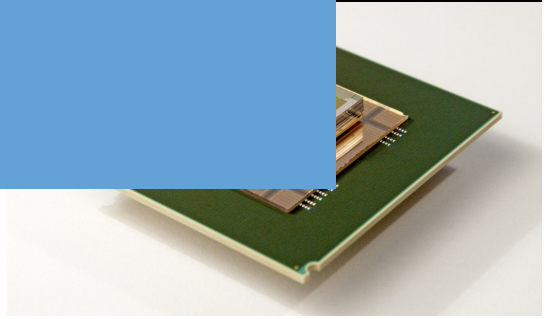


LANL – low-level interface
Terry Jones ORNL & Michael Jantz UTK a high-level interface via active profiling for data placement
Maya Gokhale LLNL higher-level Graph App interface
Ada Gavriloska GaTech a high-level interface, app characterization, ML approach
Frank Mueller NC State a high-level interface using static analysis in the LLVM layer
Si Hammond simulation of hierarchies in SST



Introspect Memory in a portable manner





Overall approach & preparation for exascale platforms.

- Describe your overall approach followed by specific activities in preparation for exascale platforms.
  - **Provide abstraction for Heterogenous memory for runtimes and applications**
    - focusing on arenas for data structures that are used together
    - Trying for inclusion in existing open source projects rather than starting a new one (CLANG, OpenMP, hwloc, Umpire, Jemalloc …)
  - **Pre-exascale environments you are using.**
    - Sierra/Summit nodes P9+Volta,
    - Intel CascadeLake + Optane,
    - Intel KNL with MCDRAM,
    - Intel w/ GPU, AMD w/ GPU -- Builds in the Aurora environment. (last week)
      - **Only** supporting unified memory architectures

*O*verall approach & preparation for exascale platforms.

- Describe your overall approach followed by specific activities in preparation for exascale platforms.
  - Include status and any results from pre-exascale environments (GPU porting, use of Summit) that illustrate your strategy.
    - Kripkie evaluation with Umpire and SICM on KNL, Sierra
    - Memsys paper on performance of apps/miniapps with SICM/Optane. VPIC showed approximately equal performance DRAM to Optane.
  - Include discussion of your major performance challenges.
    - Slow move pages on Linux kernel – (collaboration with RIKEN), working on FY20

Status of integration efforts

- Describe your overall approach and describe the status of your client integration efforts.
  - Include a description of your overall strategy for your L4 project.
    - OpenMP/CLANG/LLVM, Pull request is being updated to latest version.
    - Umpire, Pull request is there. Need to coordinate with Beckensale.
    - Hwloc -- need to push back the memory identification for numa nodes back to maintainer
  - Include specific client interactions that illustrate your strategy.
    - Umpire and OpenMP, previously Global Arrays (they have integrated our low-level allocator)
  - Include recent integration progress.
    - Umpire and OpenMP/CLANG
  - Include discussion of your major integration challenges.
    - Just being included in the runtime isn't enough to impact applications. Applications have to use the heterogenous memory in a way that increases performance. Project needs to help a few apps make use of heterogenous memory and help reason about the tradeoffs. The work in the high-level interface will inform this and is also an opportunity for an application or two to try the semi-automated interface.

PMR projects only: Specific questions to address

- Perlmutter, Aurora and Frontier represent very different node programming environments. As appropriate, how is your project addressing this challenge, including node performance, performance portability, performance of MPI+X and preparation for further heterogeneity?
  - Support of heterogenous memory as a discoverable NUMA node seems like a low bar for vendors to support.
  - Performance is more hand-tuned if the low-level interface is used. The high level-interface is working on automatically identifying data objects to move to higher performance memories.
- What is the path for your capabilities to realize sustainability in the software ecosystem?
  - Pull requests to existing opensource projects. At most have to maintain a thin wrapper for SICM.
    - Hwloc for memory discovery
    - Jemalloc for some sicm allocation features
    - Umpire for a higher level interface
    - OpenMP/CLANG/LLVM code generation

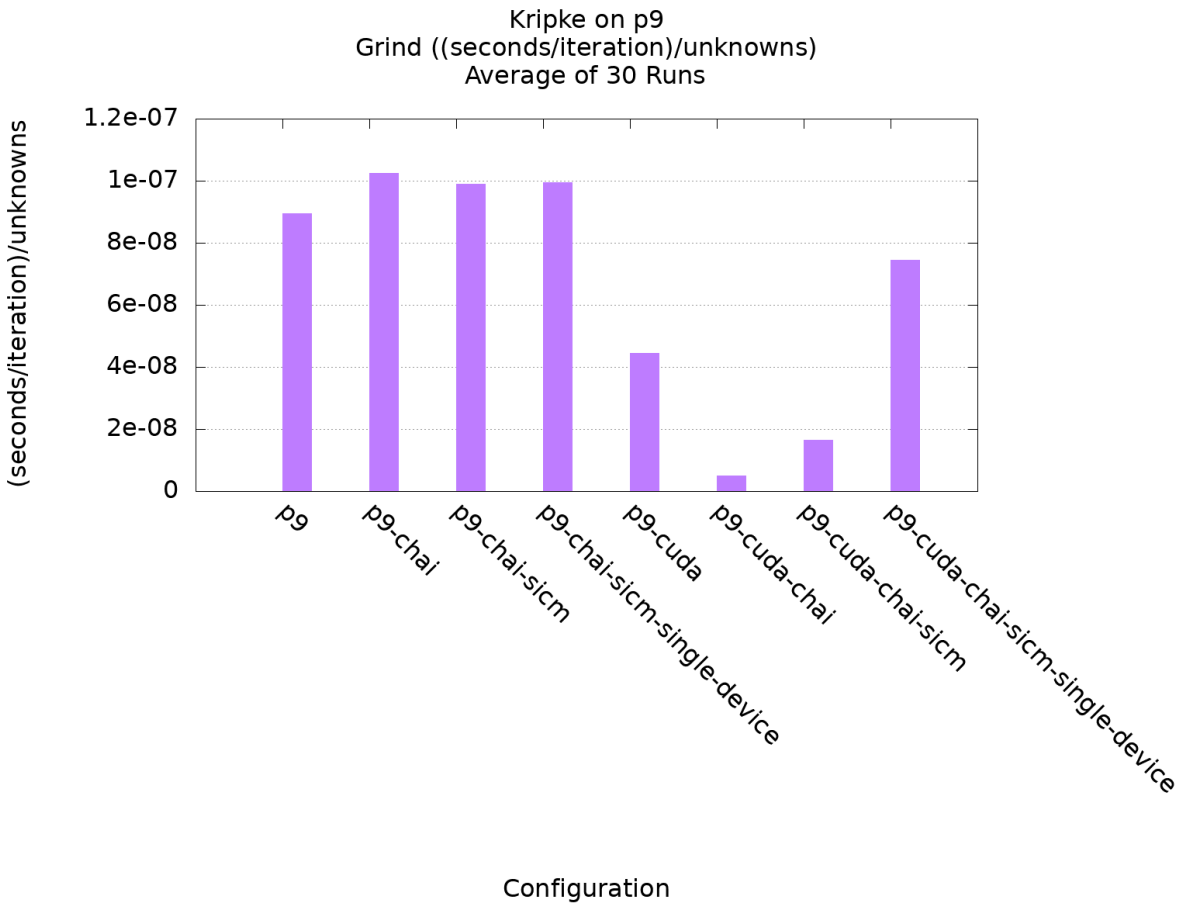
PMR projects only: Specific questions to address

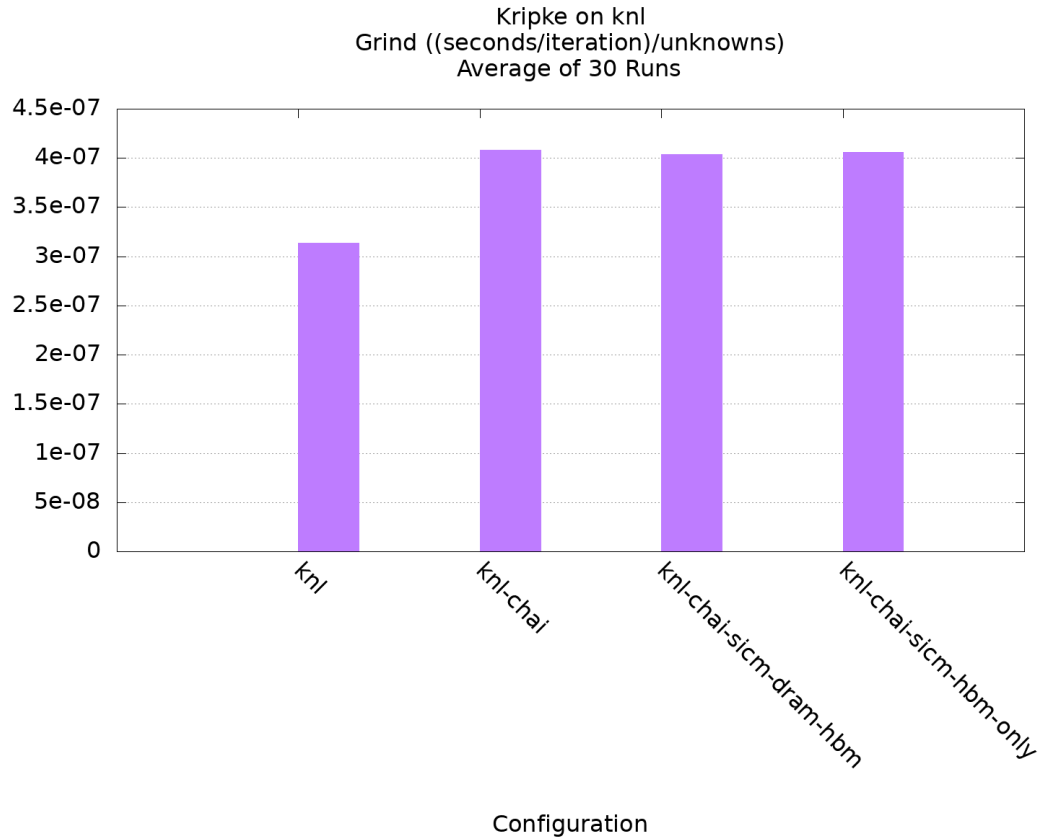
- How is your work impacting vendor capabilities?
  - Aurora: Talking to Jeff Hammond, on strategies for Aurora.
  - Perlmutter: probably will not work do to lack of unified memory on the accelerators
  - Frontier: Need more detailed hardware information – Signed up for staff for a programming workshop and will rely on Terry and the ORNL folks.

SICM low-level: Umpire + SICM Integration

- HOST and CUDA allocators implementations replaced with SICM calls
  - Devices are selected automatically
- Custom SICM operations for copy, move, memset, and realloc
  - SICM move actually moves instead of allocating new memory and copying the data
- SICM Strategy allows for explicit device selection
- Enable with CMake flags
- Benchmarks with LLNL/Kripke
  - Better than Umpire-only
  - Worse than Umpire+CUDA

Overall approach & preparation for exascale platforms.





SICM low-level: SICM initial investigation of Intel's 3DXpoint & update of SICM build system

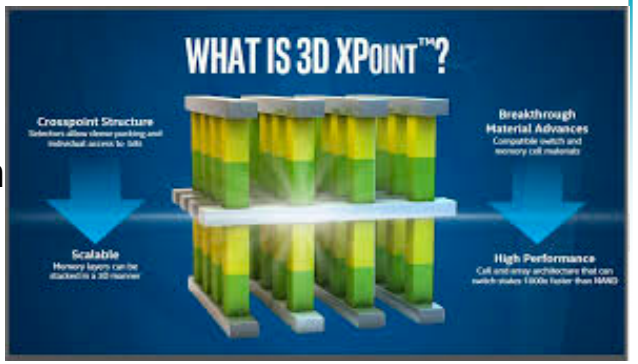
ECP WBS	2.3.1.16 SICM
PI	Michael Lang, LANL
Members	LLNL, ORNL, SNL

Scope and objectives

- Testing of 3dxpoint which is expected on A21
- Rework of build system to use Cmake and Space and Travis for continuous integration.
- To get experience on A21 hardware and to realign build system with ECP common practices.

Cool image

Intel 3Dxpoint allows Non-Volatile memory to be mapped into traditional DRAM space allowing much high memory capacity for applications but at a lower performance.



Impact

- Experience with 3dxpoint will allow support by SICM library.
- Rework of build system will allow easier integration into ECP SDK. Cmake and Spack.

Project accomplishment

- Availability of 3Dxpoint lifted the priority of this milestone.

SICM low-level: Added SICM support for Intel Optane

- Gained Experience with Intel's Optane DC PMM which is on Aurora.
- Used kernel modification to use Optane in unsupported way
  - Can also use a kernel parameter.
- Memsys 2019 - Performance characterization of a DRAM-NVM hybrid memory architecture for HPC applications using Intel Optane DC Persistent Memory Modules
  - Ran many different HPC applications using Optane
  - Compared runs in Memory Mode and Hybrid Mode
- HPDC - Using Non-Volatile Memory in High Performance Computing to Shrink the Size of Clusters
  - Showed that the size of Optane allows for applications running on a single node to have comparable performance to multimode runs

Optane Performance Evaluation

- HPC Applications and mini-apps
  - AMG, LULESH, VPIC and SNAP
  - Comparing DRAM-only (Flat mode), Optane-only (Flat mode) and Memory mode
  - Small to medium problem size
  - Strong and Weak scaling using MPI
    - Execution time & bandwidth
    - Execution time & Energy
    - L3 cache miss ratio & Cycles/Instructions
  - Used LIKWID to collect the statistics

SICM low-level: Heterogenous Memory Identification

NUMA Node Characterization

- Small C program
- Runs a few kernels to get timings for different access patterns
- Clusters results by NUMA node
  - K-Means
- Assigns type according to characteristics of cluster
- Takes 1-30 seconds (slower on Optane), depending on memory types being benchmarked

SICM low-level: Generate SICM call from OpenMP Pragmas

Patches to Clang to turn OpenMP memory spaces in OpenMP 5.x into sicm library calls in the LLVM/OpenMP runtime.

Supports Compile SICM runtime with CLANG/LLVM

At init time it does DLOpen to find SICM library, if found it it uses SICM to satisfy the

pragma openmp allocate

OpenMP memory types: *omp_ (default, large_cap, const, high_bw, low_lat) _mem_spaces*

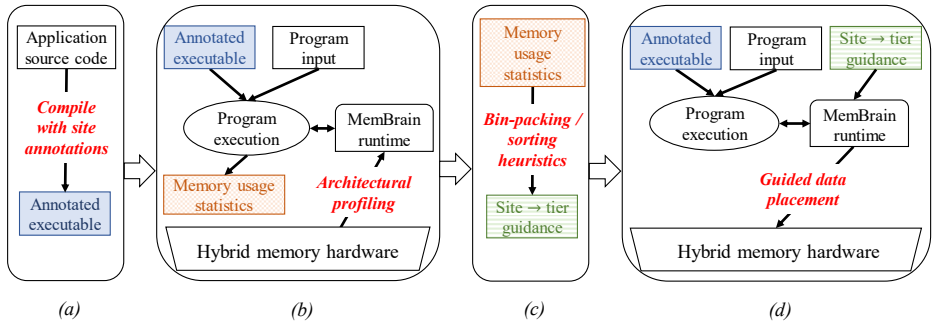
Currently supports KNL, Optane, testing Sierra

Same codepath that supports memkind library, refactored to support multiple custom memory allocators – more general than SICM support.

SICM High-level: Portable Application Guidance for Complex Memory Systems (MemSys '19)

Application Guided Data Tiering in SICM

- Extended SICM high-level interface with application-directed data tiering based on the MemBrain approach (Olson et al., 2018)

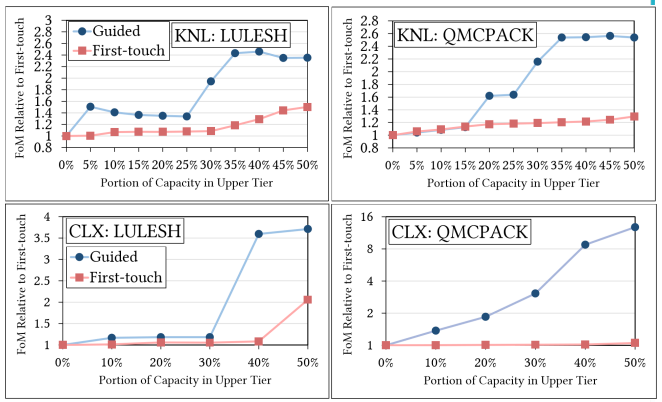


Evaluation

- Experiments on two real heterogeneous memory platforms:
  - KNL with 16 GB of high-bandwidth MCDRAM, 96 GB of DDR
  - CLX with 192 GB of DDR, 512 GB of non-volatile AEP
- Workloads from CORAL benchmark suite
  - LULESH, SNAP, AMG, and QMCPACK
  - Tested multiple inputs of each ranging from SMALL (requires only a few GB of data and only a few minutes of run time) to LARGE or HUGE (requires almost all memory capacity and several hours of run time)
- Comparison configurations
  - First touch: unguided software-based tiering
  - Cache mode: hardware manages upper tier as a large memory-side cache

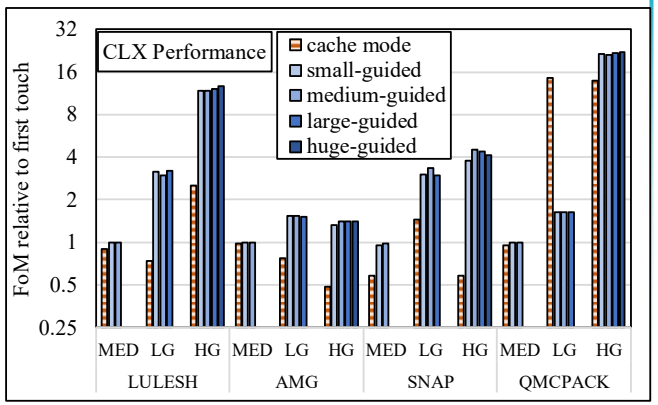
Performance with Different Upper Tier Capacities

- Guided approach is more effective than unguided first touch regardless of the upper tier capacity
- Benefits are more pronounced for configurations with more upper tier capacity
- Guidance is more important on CLX due to limited bandwidth of lower tier



Performance with Different Program Inputs

- Profiles of small program inputs are often effective for guiding execution with larger inputs
- Guided execution in SICM outperforms default first touch and cache mode by up to *22x* and *7.8x*, respectively



SICM high-level: Evaluating the Effectiveness of Program Data Features for Guiding Memory Management (MemSys '19)

Program Data Features for Guiding Memory Mgmt

- SICM may employ program profiling and analysis to direct data management across complex memory hierarchy
- Naïve strategies for collecting and using memory management guidance are ineffective due to large number of addresses and accesses generated by most applications
- Potential solution: associate profiles of memory usage with *program data features* (e.g., object sizes, types, allocation instructions, etc.)
- Problem: which data features are most effective for guiding complex memory management?

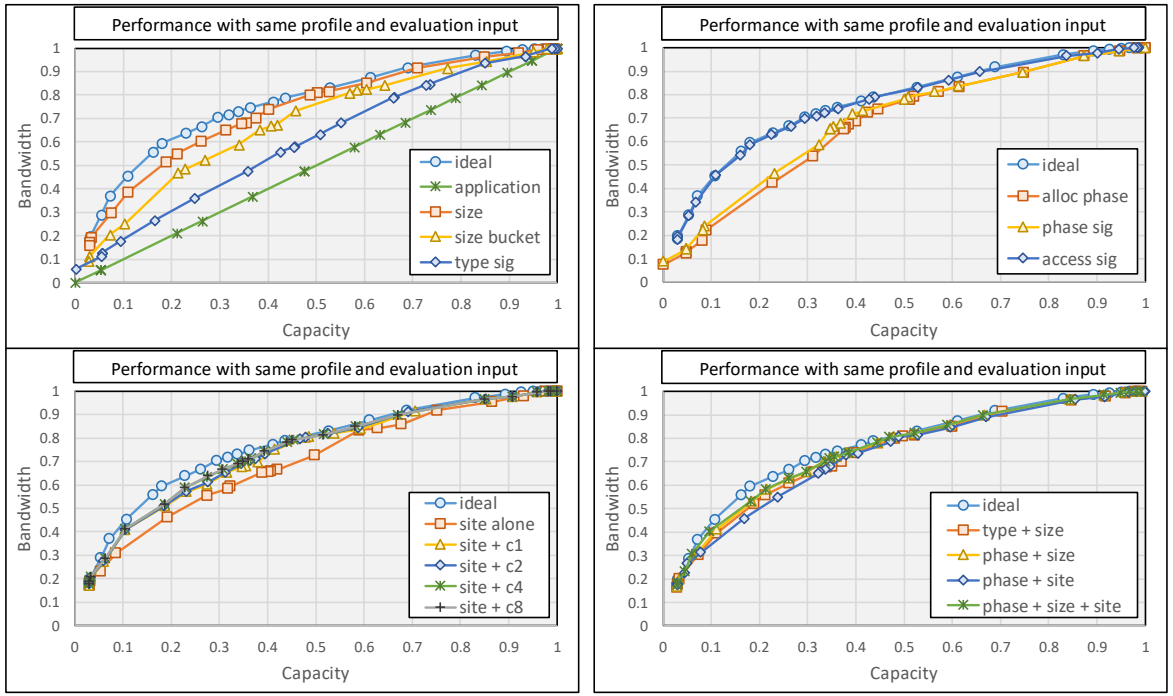
Full Program Simulations of Object Usage Behavior

- Simulated memory usage of individual data objects, including capacity, bandwidth, cache utilization, and lifetime
- Associated and aggregated usage information for each allocated object with program data features

Feature	Distinguishes objects …
application	in the same application
size	allocated with exactly the same size
size bucket	allocated with similar, but not necessarily identical sizes
type	with the same data type
allocation phase	allocated during the same phase
phase signature	alive during the same set of phases
access signature	accessed by the same set of instructions
allocation site	allocated from the same instruction
allocation context	Allocated from the same calling context

Case Study: Guided Data Tiering

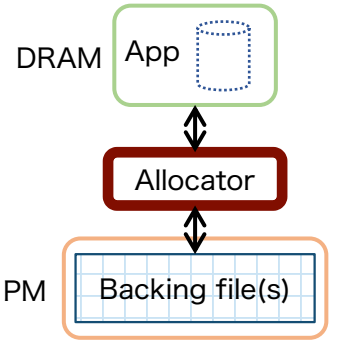
- Modeled impact of using different data features to steer hot program data into capacity-constrained device tier
  - For each feature category, classify objects associated with different features into different feature sets
  - Plots show cumulative bandwidth and capacity of data associated with hottest feature sets from left to right
- Conclusions
  - Even simple program features (e.g., object size) are often effective when profiled and guided execution use same input.
  - Allocation sites are still effective when profiled and guided execution use different inputs



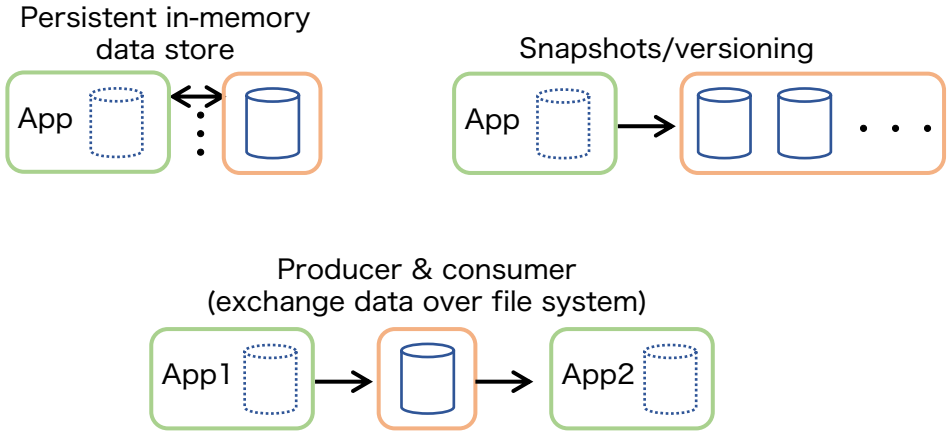
SICM LLNL task: Metall, Meta Allocator for persistent memory

Overview

- Enable applications to allocate data including custom C++ data structures in persistent memory (PM)
- Provides rich C++ API developed by Boost libraries to increase usability
- Works on both conventional block-storage and emerging byte-addressable PMs for portability
- Incorporates state-of-the-art allocation algorithms to scale to exascale
- Provides space-efficient ("diff" based) persistent memory snapshotting (versioning) capabilities to handle exascale data



Potential use cases

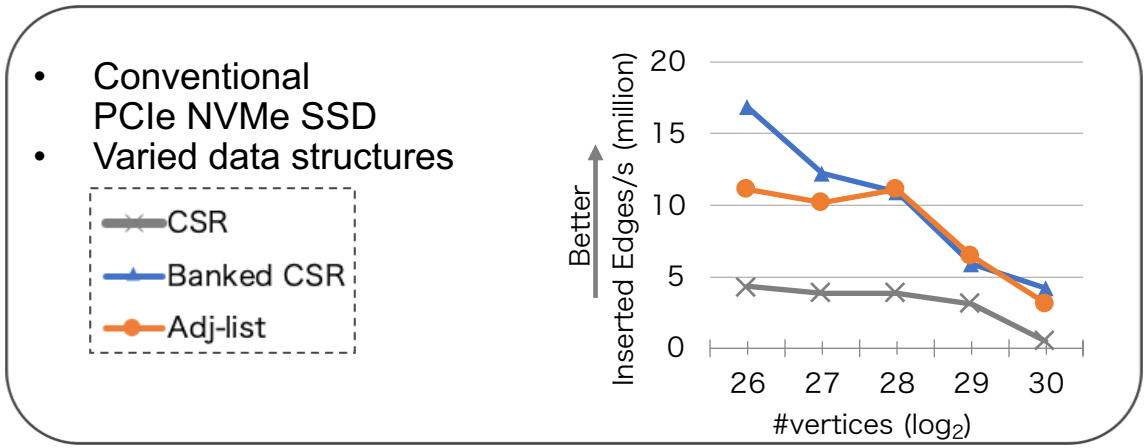


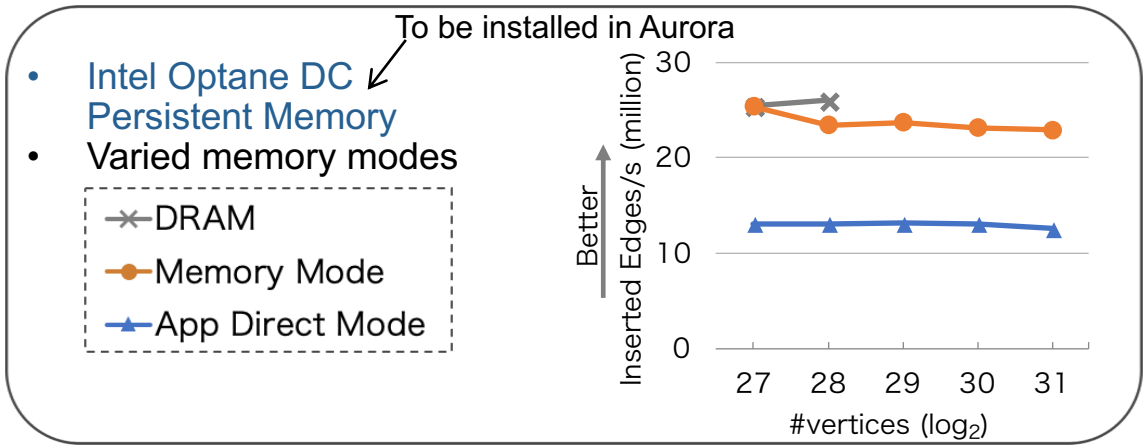
Example: allocating a custom data structure with Metall

```
class my_class {int n;}

{
  metall::manager mgr(metall::create_only, "/ssd");
  auto pdata = mgr.construct<my_class>("data")();
  pdata->n = 10;
}

// -- Exit the program and reattach the data -- //

{
  metall::manager mgr(metall::open_only, "/ssd");
  auto pdata = mgr.find<my_class>("data").first;
  pdata->n += 20; // Can update data
}
```

Metall : Meta allocator for persistent memory

- Graph construction (write intensive) benchmark (primary evaluation for collaboration with EXAGRAPH)





Metall enables applications to process exascale data
in **a variety of conditions** (PM technologies and custom data structures)
with **straightforward modifications**

- Client integration progress
  - Metall runs on commodity Linux systems and requires only Boost libraries
    - XFS, ZFS, or Btrfs filesystem is necessary for space-efficient snapshotting
  - Potential collaboration
    - **EXAGRAPH** Collaborating to store graph data as well as other intermediate data into PM leveraging Metall.
    - **EXAALT** Investigating a collaboration opportunity. Passed an initial unit test to store its management data.